

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de ciencias y sistemas
Arquitectura de computadores y ensambladores 2
Sección D
Primer semestre 2025



Proyecto: Fase 3

Estudiante

Diego Felipe Cali Morales
Bruce Carbonell Castillo Cifuentes
Natalia Mariel Calderón Echeverría
Daniel Eduardo Velasquez Avila

Carnet

202201128
202203069
2022200007
202200041

Índice

Prototipo - Módulos	3
Motores	9
Base de datos	11
Maqueta	12
Diseño final (Fase 3)	14
Dashboard	15
Grafana Dashboard	16
Predicción	18
Stack IoT Framework	19
1. Capa de Percepción (Sensors)	19
2. Capa de Red (Connectivity)	19
3. Capa de Procesamiento (Analytics)	20
4. Capa de Aplicación (Smart Apps)	20
5. Capa de Infraestructura de Producto (Product Infrastructure)	20
Beneficios del Sistema	21
Impacto Ambiental	22
Presupuesto	23
Bocetos	24
Link del repositorio	25

Prototipo - Módulos

- **Sensor ultrasónico**

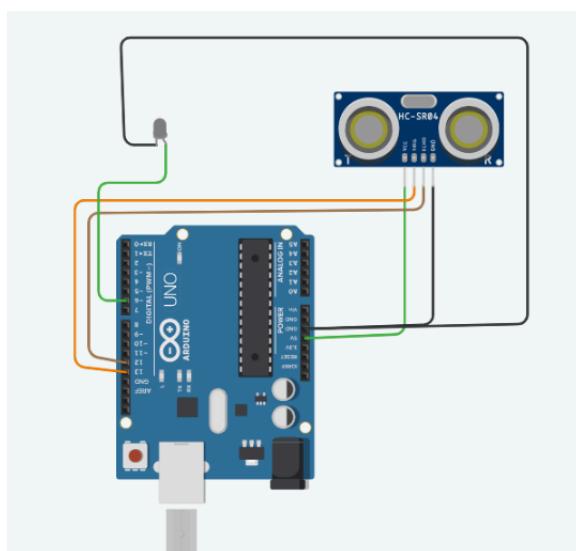
El uso del sensor ultrasónico se debe a la necesidad de detectar la presencia de una persona, esto mediante la medición de la distancia a la que se encuentra del propio sensor. Primero se asignaron los pines específicos para el LED y para el sensor ultrasónico, donde TRIG envía la señal de disparo y ECHO recibe el eco que indica la distancia.

```
int TRIG = 13;  
  
int ECHO = 12;
```

Posteriormente, en el área de setup se establecen los modos de entrada y salida para cada pin y se inicia la comunicación serial para la visualización de datos.

```
void setup(){  
  
    pinMode(TRIG, OUTPUT); //Salida  
    pinMode(ECHO, INPUT); //Entrada  
  
    pinMode(LED, OUTPUT);  
    Serial.begin(9600);  
}
```

En el loop, se activa el pin de Trigger para enviar un pulso ultrasónico muy breve. Luego, se mide el tiempo que tarda en recibir el eco mediante pulseIn en el pin ECHO. La distancia se calcula dividiendo el tiempo de eco entre 58.2 (para convertirlo a centímetros).



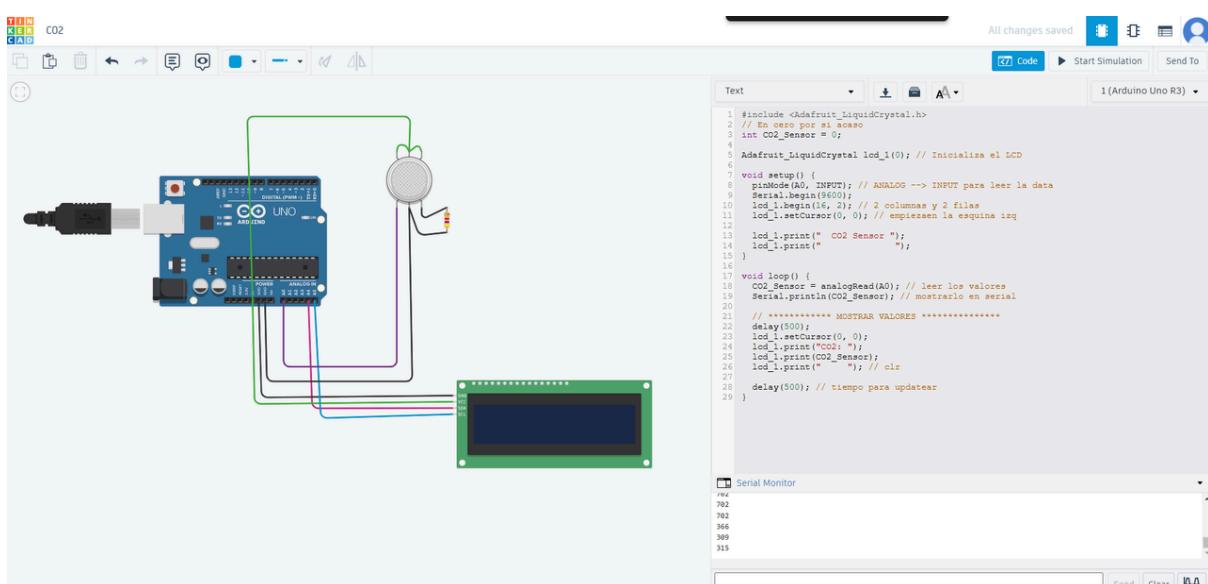
- **Sensor CO2**

En este caso se diseñó un circuito pude manejar un display LCD y mostrar la lectura de dicho sensor. Primero se establece la librería necesaria para controlar la LCD.

```
#include <LiquidCrystal_I2C.h>
```

Luego es necesario declarar una variable, en donde se almacenará la lectura analógica proveniente del pin analógico al que se encuentra conectado. En el setup, es necesario declarar el pin analógico como un pin de entrada para posteriormente poder leer los valores en el Loop.

```
CO2_Sensor = analogRead(A0);
```



- **Monitoreo Lumínico**

En este caso se buscaba añadir el monitoreo lumínico al funcionamiento, para esto al inicio se asignan los pines y variables necesarias.

```
const int pinLDR = A1;
luz (LDR) int valorLDR = 0;
```

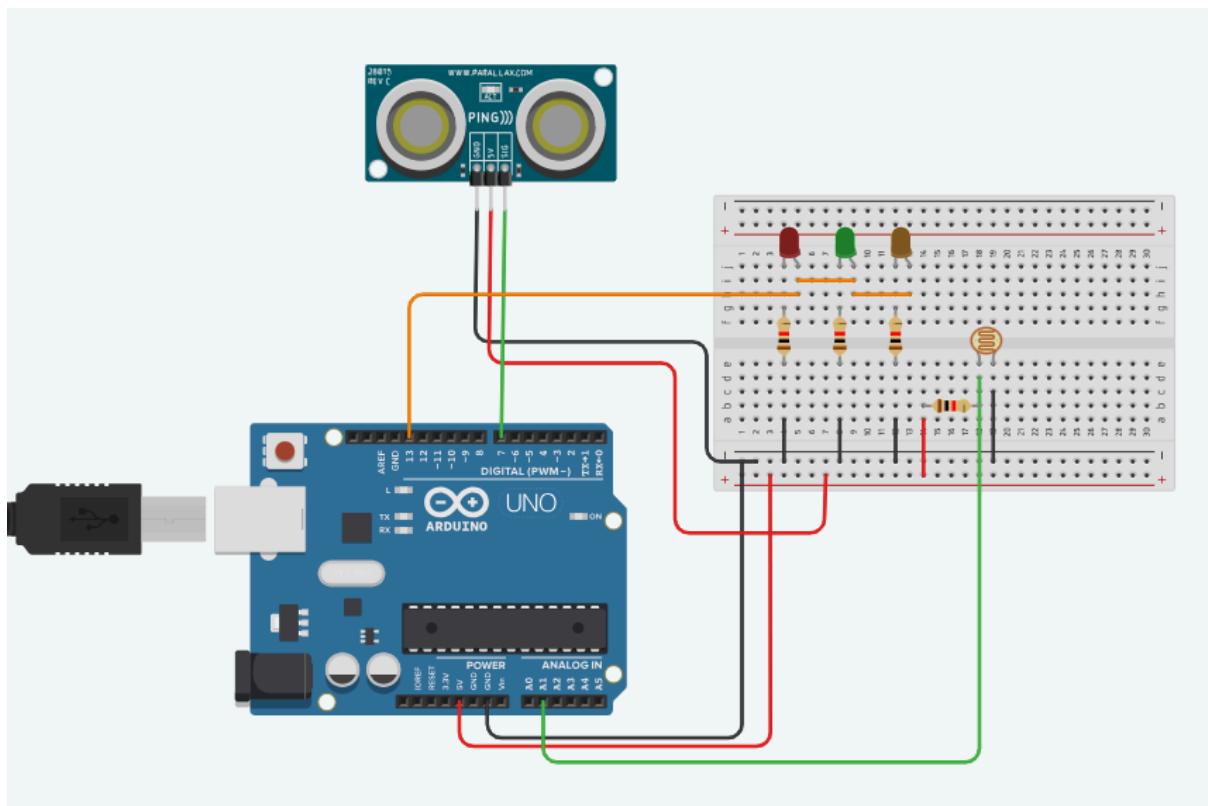
Posteriormente, se define en la función functionController la lectura de dicho sensor y cuando ya se haya leído, el valor se imprime en el monitor serial junto con la medición de distancia:

```

valorLDR = analogRead(pinLDR);
Serial.print("Luz: ");
Serial.println(valorLDR);

```

Por ultimo, valor del sensor de luz se utiliza para determinar si el ambiente tiene "poca luz", si la intensidad de luz es baja ($\text{valorLDR} \leq 500$) y se detecta un objeto a 200 cm o menos, se activan los LED



- **Sensor de temperatura y humedad**

Para el monitoreo de temperatura y humedad al inicio se define el pin y el tipo del sensor DHT22, y se crea un objeto dht para gestionar las lecturas.

```

// Configuración del sensor DHT22
#define DHTPIN A3
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

```

Luego se obtienen los valores de dichas lecturas y se establece el manejo de errores en caso de que alguna lectura no sea válida.

```

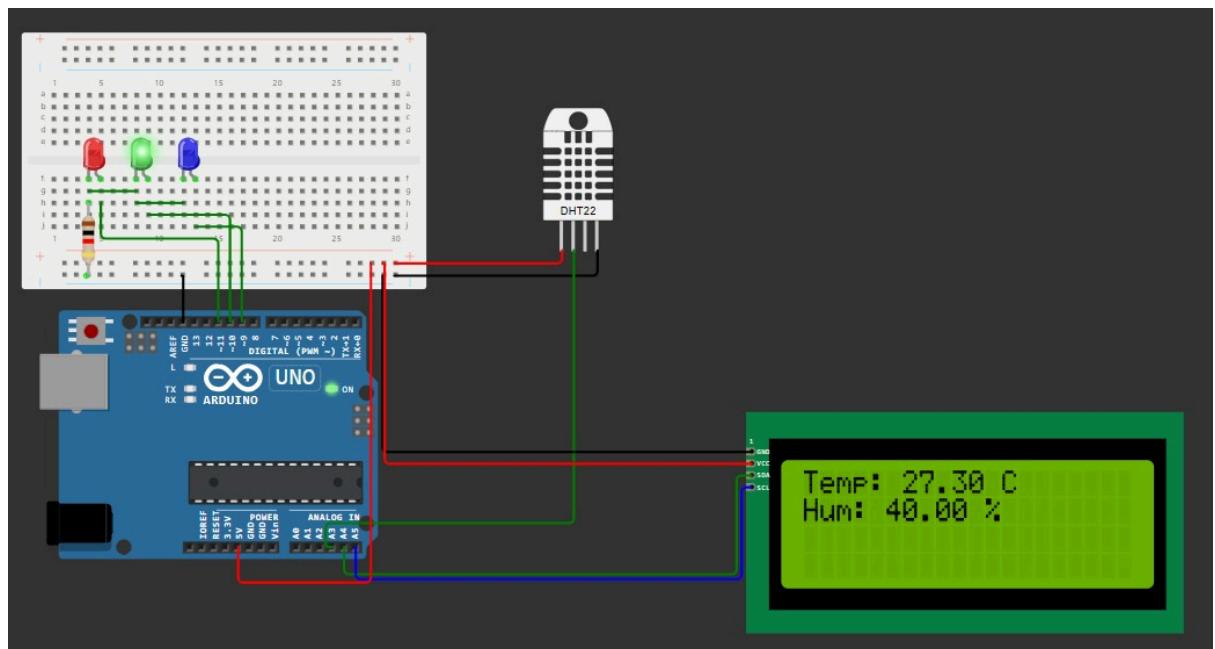
float humedad = dht.readHumidity();
float temperatura = dht.readTemperature();

// Verificar si la lectura es válida
if (isnan(temperatura) || isnan(humedad)) {
    lcd.setCursor(0, 0);
    lcd.print("Error leyendo");
    lcd.setCursor(0, 1);
    lcd.print("DHT22...");
    delay(2000);
    return -1;
}

```

Una vez validados, se muestran en pantalla los valores actuales de temperatura y humedad. Según la temperatura medida se cada una de las distintas LEDs.

- $\geq 30^\circ\text{C}$: Se enciende el LED rojo para indicar una temperatura alta.
- $\leq 18^\circ\text{C}$: Se enciende el LED azul para indicar una temperatura baja.
- Entre 18°C y 30°C : Se enciende el LED verde para indicar una temperatura moderada.



- **RFID RC 522**

Los lectores RFID (radio frequency identification) consisten en el uso de un tag. El tag tiene la capacidad de enviar información al lector RFID, este código puede ser usado para autorizar permisos o para activar ciertos actuadores.

Para este proyecto este sensor ayuda a validar únicamente la entrada de personal autorizado al cuarto de servicio. La llave para poder ingresar puede ser el tag en cualquiera de sus modalidades: llavero o tarjeta.

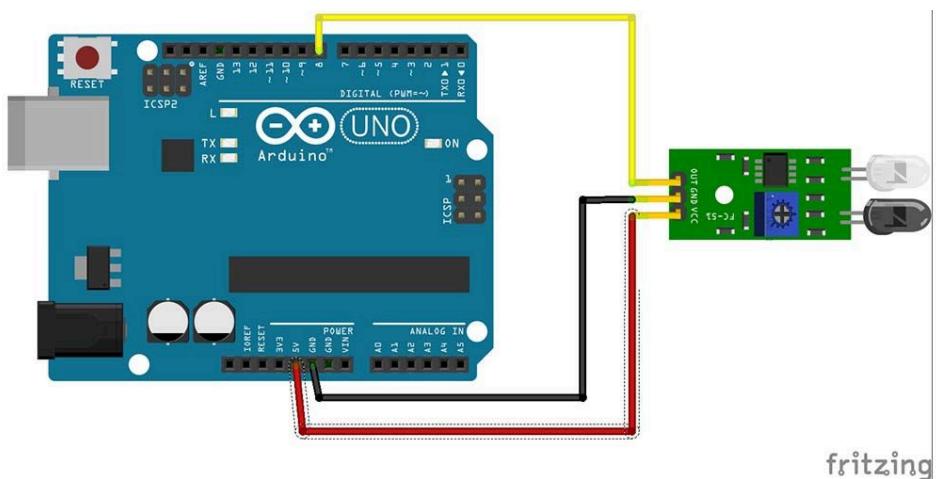
PIN	Arduino PIN
SDA	53
SCK	52
MOSI	51
MISO	50
RST	5
GND	GND
VCC	3.3



- **Sensor Infrarrojo:**

El sensor infrarrojo de 3 pines en Arduino generalmente tiene tres conexiones: VCC, GND y OUT. El pin VCC se conecta al suministro de 5V del Arduino para alimentar el sensor, mientras que el pin GND se conecta a tierra para completar el circuito. El pin OUT es donde se obtiene la señal del sensor, la cual puede ser digital o modulada, dependiendo de su tipo y función.

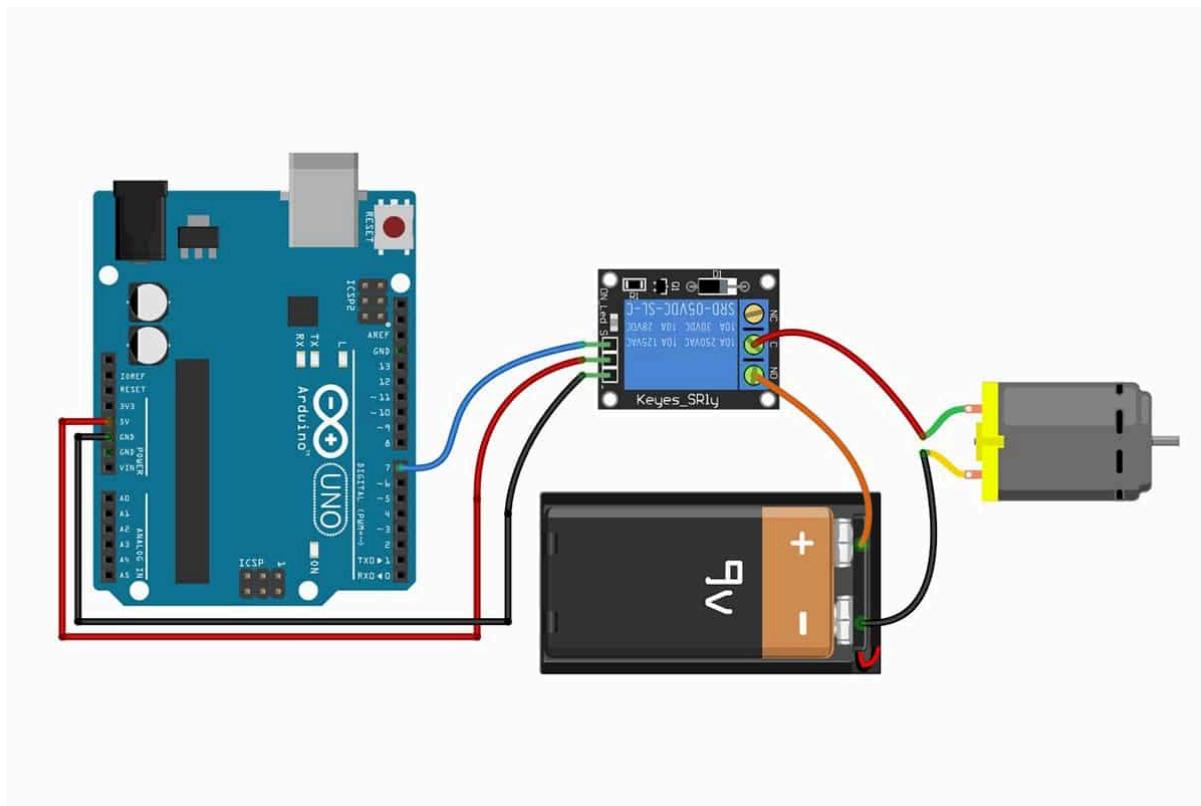
Si el sensor es un sensor de distancia infrarrojo, el pin OUT generalmente emite una señal digital que indica la presencia de un objeto cercano, enviando un valor HIGH o LOW. Para sensores de emisión y recepción IR, el pin OUT puede ser utilizado para la transmisión o recepción de señales de control remoto. En ambos casos, Arduino puede leer esta señal y utilizarla en diversas aplicaciones como detección o comunicación.



Motores

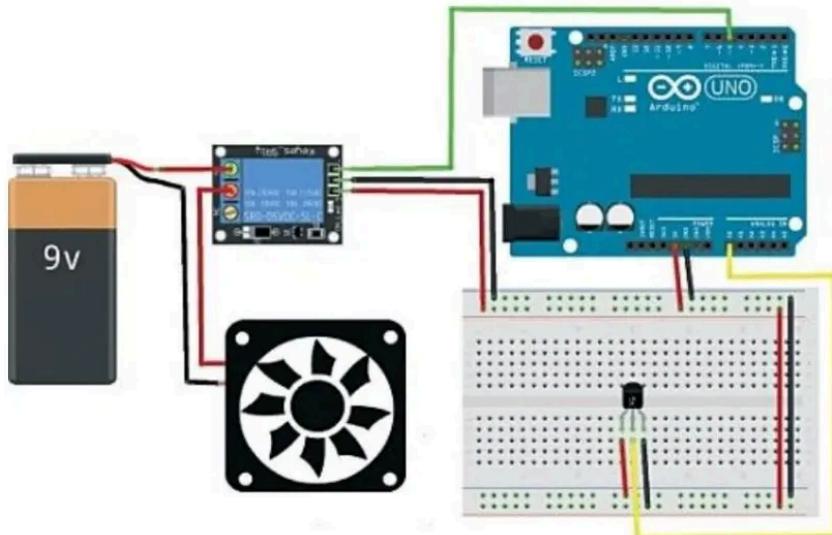
- **Motor DC:**

Un motor DC convierte la energía eléctrica en movimiento mediante la interacción de un campo magnético y un rotor. Cuando se conecta a un Arduino, este proporciona corriente a través de un transistor o un relé, pero debido a que el motor necesita más corriente de la que el Arduino puede suministrar directamente, se utiliza un relé para controlar el flujo de electricidad. El relé actúa como un interruptor que permite que el Arduino active o desactive el motor sin sobrecargar sus pines. Esto asegura que el motor reciba la corriente adecuada sin dañar la placa de Arduino.



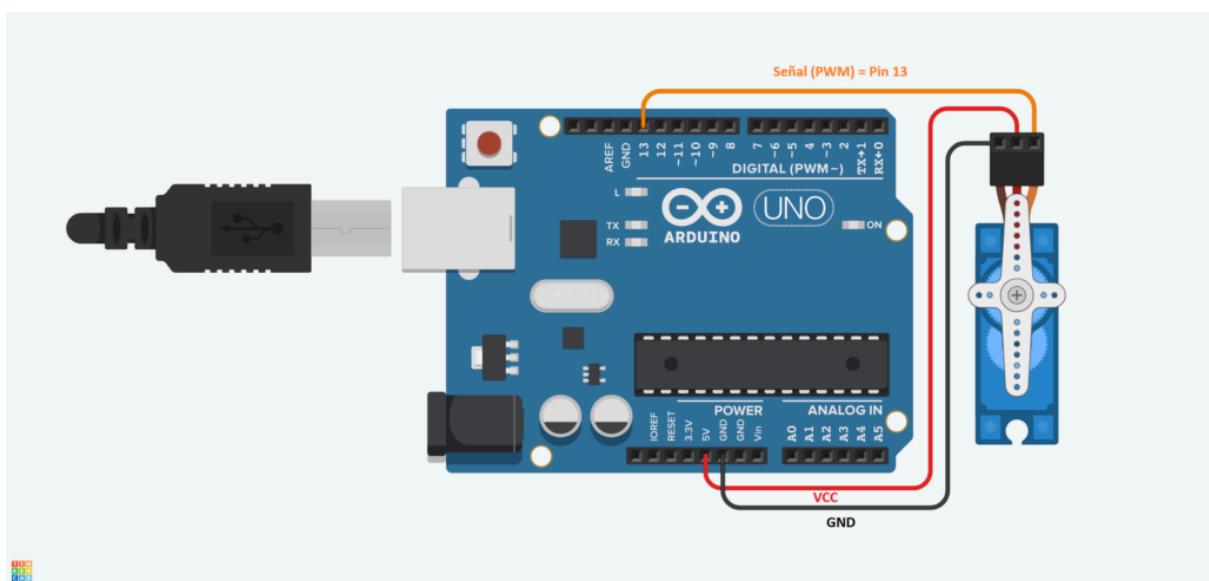
- **Ventilador:**

Un ventilador de 5V funciona de manera similar a un motor DC, convirtiendo la energía eléctrica en movimiento para generar aire. Para controlarlo con Arduino, se utiliza un relé, ya que el ventilador generalmente consume más corriente de la que un pin de Arduino puede manejar directamente. El Arduino activa el relé, que a su vez cierra el circuito y permite que el ventilador reciba la corriente adecuada. Esto protege la placa de Arduino y asegura que el ventilador pueda encenderse y apagarse de manera segura sin sobrecargar el sistema.



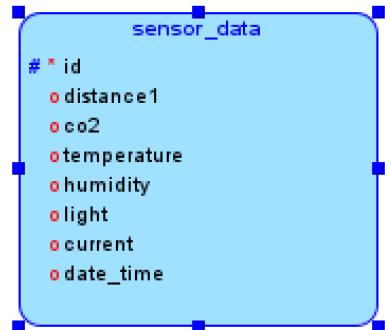
- **Servomotor:**

Un servomotor es un motor de corriente continua diseñado para moverse a posiciones angulares específicas. A diferencia de los motores DC comunes, el servomotor tiene un sistema de control integrado que le permite ajustar su ángulo con precisión, generalmente entre 0° y 180°. Esto se logra mediante una señal PWM (modulación por ancho de pulso) enviada por un microcontrolador, como Arduino, que controla la posición del motor ajustando el ciclo de trabajo de la señal. Los servomotores son ampliamente utilizados en aplicaciones que requieren movimiento preciso, como robótica, sistemas de control de cámaras y automatización.



Base de datos

- Modelo Entidad Relación

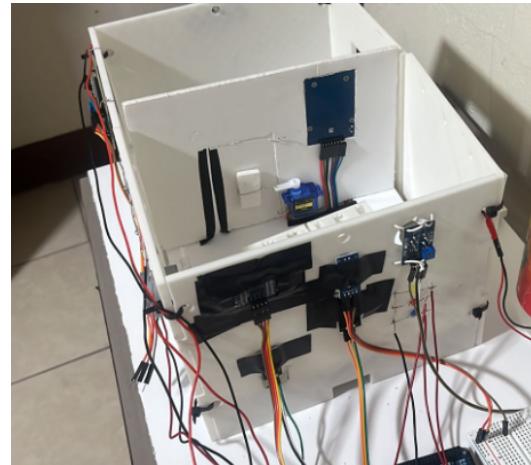


sensor_data	
P	* id
	INTEGER
	distance1
	FLOAT (10)
	co2
	FLOAT (10)
	temperature
	FLOAT (10)
	humidity
	FLOAT (10)
	light
	FLOAT (10)
	current
	FLOAT (10)
	date_time
	DATE
sensor_data_PK (id)	

Maqueta

El objetivo principal de la maqueta consistía en desarrollar un Sistema Inteligente de Monitoreo Ambiental para Cuartos de Servidores mediante IoT. Por ello, se estableció que la maqueta debía ser lo suficientemente espaciosa para que la presencia de los sensores no redujera el espacio disponible ni afectaría la estética visual. Tomando en cuenta los sensores a utilizar, se determinaron dimensiones grandes. A partir de ahí, se inició el proceso de diseño de los ambientes y la distribución de los sensores. La maqueta fue realizada a través de modelado 3D y posteriormente impresión 3D. Debido a que cada plancha toma alrededor de 11 - 13 horas para imprimirse completamente fue necesario planear la impresión de una plancha diaria.

Los sensores fueron colocados en lugares estratégicos para garantizar su correcto funcionamiento sin comprometer la armonía visual. Esto se consideró desde la fase de modelado 3D. Se buscó que su presencia pasara lo más desapercibida posible, manteniendo un equilibrio entre funcionalidad y estética. **Así comenzó el proceso de instalación de los sensores (Fase 2):**



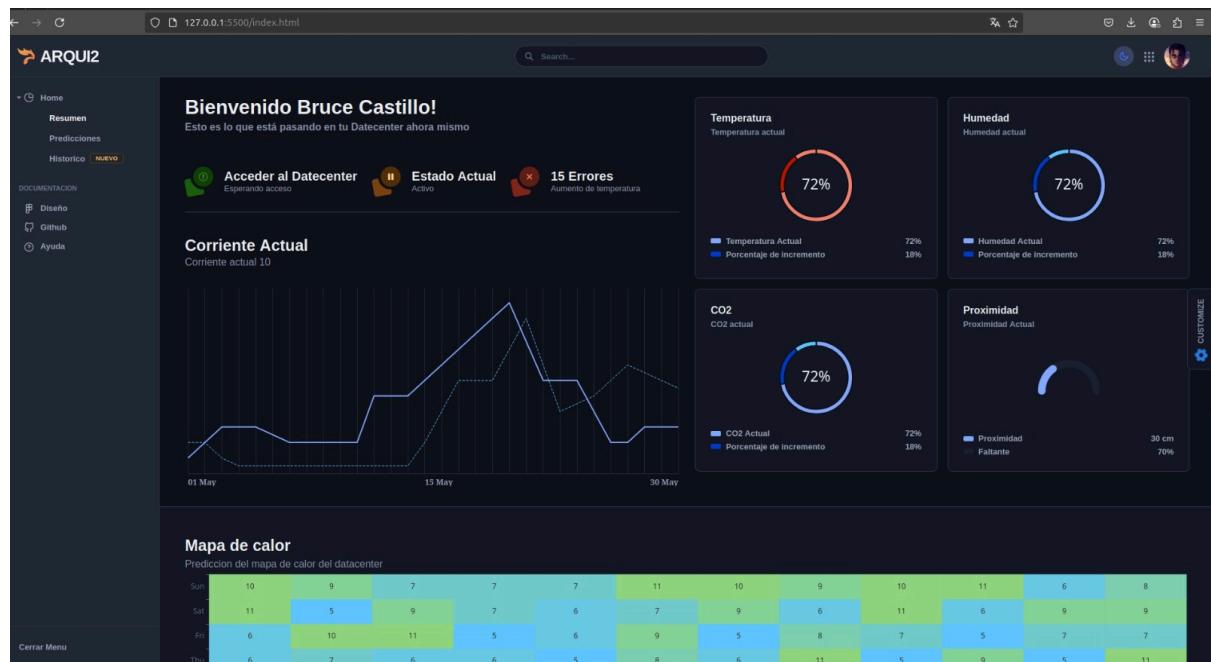
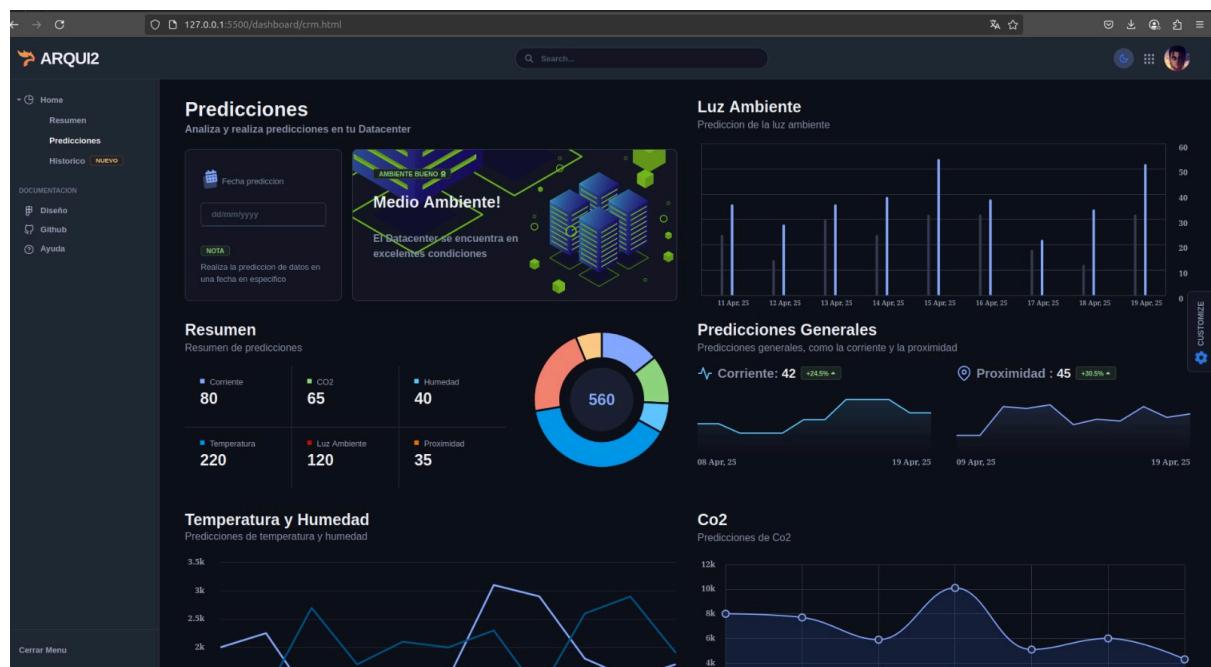


Diseño final (Fase 3):

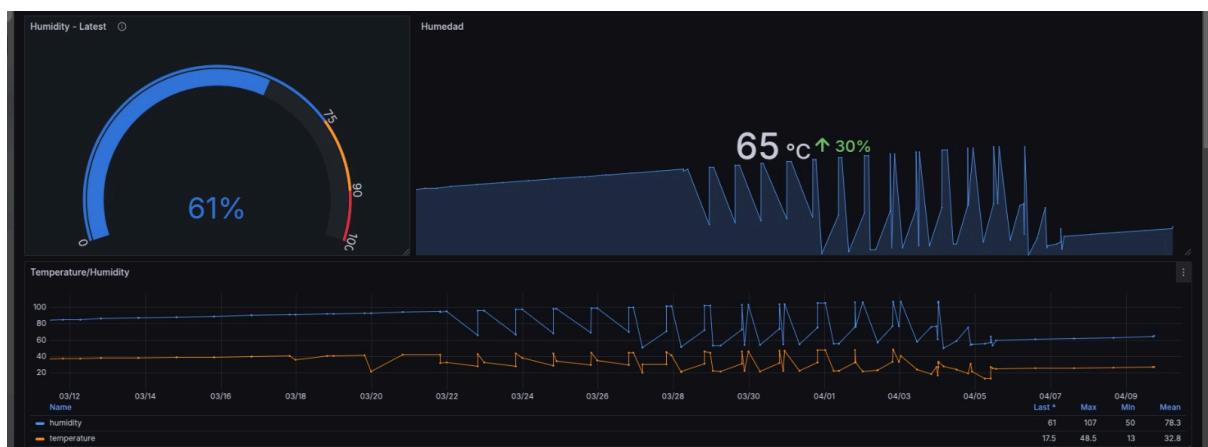
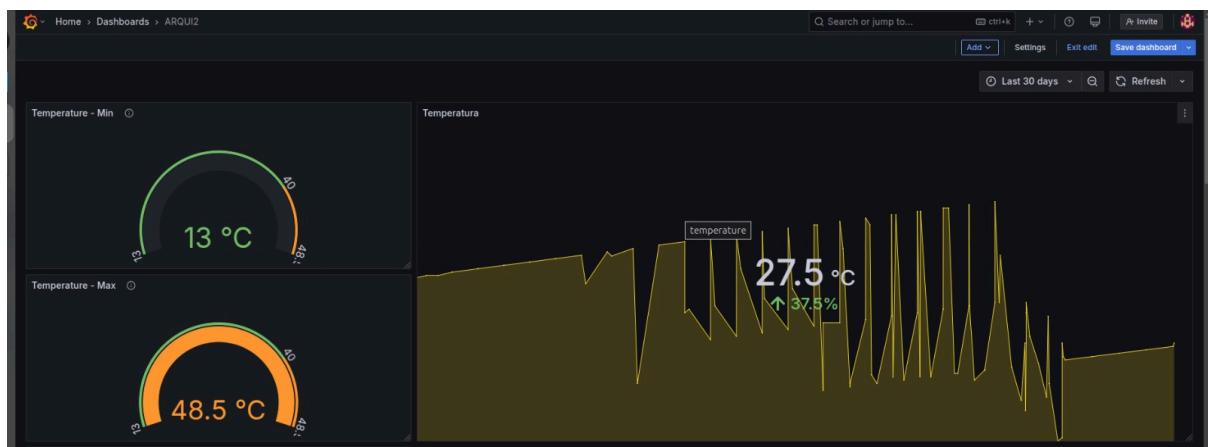




Dashboard



Grafana Dashboard





Predicción

Durante la fase 2 se propuso el modelo ARIMA para la predicción de valores, sin embargo, durante la implementación resultó un poco mas complicado y limitado debido que no modela bien estacionalidades complejas y es más complicado de configurar, que que se debe ajustar constantemente los valores p, d y q. Estas características hacen que el modelo ARIMA no sea el indicado para este proyecto, ya que el foco es la automatización y eficiencia de los procesos. Es por eso que se usó el modelo Prophet.

El modelo prophet es un modelo de series temporales desarrollado por Facebook, su diseño se enfoca en la simplicidad, alcance y eficiencia es por eso que puede llegar a ser muy poderoso y trabajar con grandes cantidades de datos sin comprometer su eficiencia. Prophet se basa en un modelo aditivo que combina tres componentes principales: tendencia, estacionalidad y efectos de días especiales. Prophet modela la serie de tiempo en base a la siguiente fórmula:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t.$$

y= modelo/predicción

g= tendencia

s= estacionalidad (ciclos semanales o anuales)

h= componente festivo (días festivos o anomalías)

et = error/noise

Una de las ventajas que nos llevó a inclinarnos a Prophet es el hecho que a pesar de trabajar con una fórmula completa, prophet únicamente requiere de dos columnas necesarias: y para valores y ds para fechas. Lo que lo vuelve ideal en este caso ya que los valores que se envian son: La fecha del valor registrado y el valor registrado (temperatura, humedad, CO2, corriente y luz). Prophet trabaja perfectamente con los datos enviados y detecta las tendencias de manera rápida y automática lo que cumple con el objetivo de esta fase. Otra de las ventajas del uso de Prophet es que a diferencia de ARIMA, no es necesario hacer pruebas de estacionalidad ni definir los parámetros de manera manual, ya que prophet se encarga de inferir de manera interna estos parámetros y los patrones.

Stack IoT Framework Stack IoT Framework

Para organizar la arquitectura del sistema, se utilizó el Stack IoT Framework, que se compone de varias capas que facilitan la interacción entre los sensores, el procesamiento de datos y la comunicación con la nube. Permite la modularidad y escalabilidad del sistema, asegurando un flujo de datos eficiente desde la captura hasta la presentación final.

Gracias a esta estructura, el monitoreo ambiental se vuelve más preciso y adaptable a diferentes condiciones y entornos. A continuación, se describen las capas involucradas:

1. Capa de Percepción (Sensors)

Esta es la primera capa del sistema y se encarga de la adquisición de datos mediante los sensores, también incluye actuadores como los motores, ventiladores y buzzer. Aquí se incluyen:

- Sensor ultrasónico para detección de proximidad.
- Sensor de temperatura y humedad DHT22.
- Sensor de CO2.
- Sensor LDR para monitoreo lumínico.
- Sensor de corriente ACS712
- Módulo RFID
- Cámara de reconocimiento facial

Estos sensores capturan información relevante sobre el entorno y la transmiten al sistema para su análisis.

2. Capa de Red (Connectivity)

Esta capa se encarga de la transmisión de datos entre los sensores y actuadores hacia el microcontrolador, la nube y la base de datos. A través del uso MQTT se permite la conectividad y transmisión de información relevante. A diferencia de las fases anteriores MQTT facilita la comunicación y se caracteriza por ser más escalable y ordenada. Los elementos del proyecto en esta fase son:

- Módulo Wi-Fi (incluido en la raspberry pi)
- Protocolo MQTT
- Broker MQTT
- Conectividad con Raspberry Pi

3. Capa de Procesamiento (Analytics)

Esta capa como su nombre lo indica, se caracteriza por el procesamiento, análisis e interpretación de los datos recopilados. La interpretación de la información recibida permite generar alertas, generar predicciones y generar graficas para la visualización de los datos. En el caso de la fase 3, que a diferencia de fases anteriores se centra más en el procesamiento y análisis de datos, esta capa representa el corazón del proyecto ya que en ella es donde se encuentra: El análisis de datos en tiempo real e histórico, reconocimiento facial a través de AWS Rekognition, proyecciones climáticas (Prophet), Uso de grafana y condiciones para la activación de ciertos actuadores.

4. Capa de Aplicación (Smart Apps)

Esta capa proporciona la interfaz con el usuario y permite visualizar e interactuar con los datos obtenidos. En esta capa se permite monitorear y gestionar el control sobre ciertos elementos del sistema, gestionar las proyecciones y visualizar la información de cada sensor. En este caso, los elementos de la capa son:

- **Dashboard (Aplicación web)**

El dashboard permite visualizar la información recopilada y permite el control de acceso a través del uso de PIN y reconocimiento facial. El dashboard por su parte también permite la gestión de las proyecciones de datos y su visualización.

- **Grafana**

Visualización gráfica de los datos históricos recopilados

5. Capa de Infraestructura de Producto (Product Infrastructure)

Esta capa abarca la base tecnológica y física necesaria para soportar el sistema IoT. También, como su nombre lo indica, la infraestructura del mismo, que en este caso se vería reflejado en la **maqueta elaborada y los sensores y actuadores utilizados**.

Algunos de los elementos de esta capa dentro del proyecto son: Arduino Mega, Raspberry Pi, Base de datos en la nube, sensores utilizados, actuadores utilizados, maqueta elaborada (infraestructura física), Base de datos en la nube, los distintos servicios de la nube utilizados, grafana, entre otros.

Beneficios del Sistema

La automatización y monitoreo constante de cualquier sistema trae en si misma beneficios inherentes como lo es el ahorro de tiempo y mayor productividad.

La fase 3 de este proyecto supone ciertos beneficios tanto al control de cuarto de servidores como a la gestión y control del mismo, entre ellos están:

- **Monitoreo en Tiempo Real**

Permite la supervisión continua de las condiciones ambientales en un cuarto de servidores, lo que permite un mejor monitoreo y control sobre el estado del cuarto de servidores y los servidores en si. El monitoreo en tiempo real también aumenta la seguridad, ya que se puede saber si hay alguien dentro del cuarto.

- **Automatización de Respuestas**

Activa alarmas o luces de advertencia en función de las mediciones, lo que supone una prevención a fallos críticos y un mejor control del funcionamiento del mismo, ya que al detectar cualquier anomalía se puede tomar acciones determinadas

- **Optimización del Uso de Energía**

La activación de dispositivos solo ocurre cuando es necesario, como es el caso de las luces y los ventiladores. Estos no se encuentran encendidos si no es necesario su uso.

- **Reducción de Riesgos**

Minimiza fallos en equipos al controlar la temperatura y la humedad, este beneficio viene de la mano con los primeros 2 beneficios mencionados y puede llegar a significar no sólo un mejor funcionamiento si no una vida más larga a los servidores y aparatos dentro del cuarto, ya que siempre se buscan las condiciones óptimas.

- **Accesibilidad Remota**

A través del uso de la nube es posible monitorear los datos recopilados en cualquier lugar y obtener predicciones sobre los mismos. Esto favorece el control y gestión.

- **Escalabilidad**

Se pueden agregar más sensores para ampliar la funcionalidad. Esta fase demostró la escalabilidad que presenta el proyecto, ya que a lo largo de las

distintas fases el proyecto ha ido escalando. Uno de los beneficios es la base de datos en la nube.

Impacto Ambiental

- **Aumento en la eficiencia energética**

A través del monitoreo constante es posible reducir el consumo energético únicamente al necesario para asegurar una funcionalidad y condiciones óptimas. Un ejemplo concreto de este beneficio es el que se mencionaba anteriormente, ciertos sensores y actuadores únicamente son activados cuando son necesarios como es el caso de la luz y los ventiladores, que su uso sea limitado y estratégico favorece un ahorro considerable de energía.

- **Mayor Durabilidad de Equipos, reducción de desechos electrónicos**

Mantener condiciones óptimas extiende la vida útil de los distintos aparatos dentro del cuarto de servidores, no solo los servidores. Una vida útil mayor significa, a largo plazo, menos consumo de aparatos que eventualmente se traduce como menos desechos electrónicos. Una mala utilización y mantenimiento de los equipos puede llegar a elevar los desechos electrónicos producidos constantemente lo que significa contaminación innecesaria al planeta.

- **Sostenibilidad**

Este proyecto hace uso de tecnologías que fomentan la sostenibilidad como lo es el uso de cloud computing ya que se evita el uso de infraestructuras físicas. En general el desarrollo de este proyecto se concentra en el desarrollo de soluciones que favorezcan el consumo mínimo de energía y a la vez mantengan su eficiencia. Esto crea un proyecto totalmente sostenible. En un plano más grande, el uso de tecnologías sostenibles favorece al medio ambiente tanto en corto como a largo plazo.

Presupuesto

Cantidad	Producto	Precio	Total
1	Pantalla LCD 1602 con I2C	Q50.00	Q50.00
1	Modulo sensor de humedad y temperatura - DHT11	Q34.00	Q34.00
1	Sensor de calidad de aire MQ-135	Q39.00	Q39.00
1	Filamento {Impresion 3D}	Q172.00	Q172.00
1	Cartón piedra	Q96.65	Q96.65
1	Servo motor SG90 - 180	Q31.00	Q31.00
1	Modulo relay estado solido; 2 canales, 1 polo, 2 tiros	Q43.00	Q43.00
1	Módulo RFID RC522	Q45.00	Q45.00
1	Modulo buzzer activo KY-012	Q18.00	Q18.00
1	Motor DC 16500 RPM	Q9.00	Q9.00
4	LED blanco	Q1.00	Q4.00
3	LED Amarillo	Q1.00	Q3.00
1	Ventilador 3007 5V DC 30x30x7 mm	Q24.00	Q24.00
Total			Q568.65

Bocetos

Diagrama Funcionamiento (Boceto fase 1 y 2)

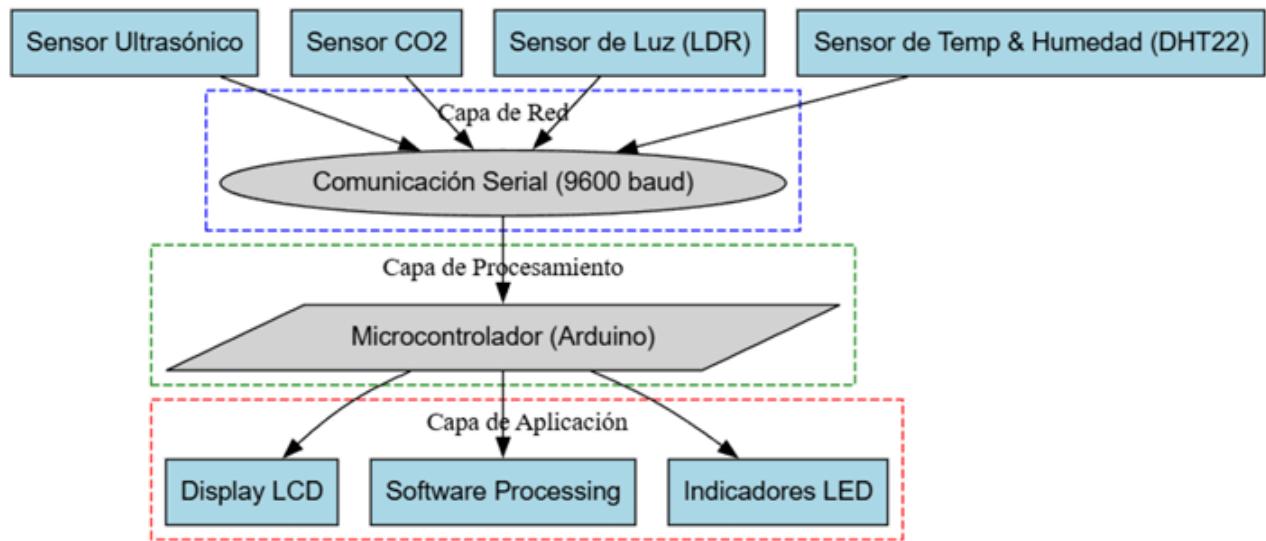
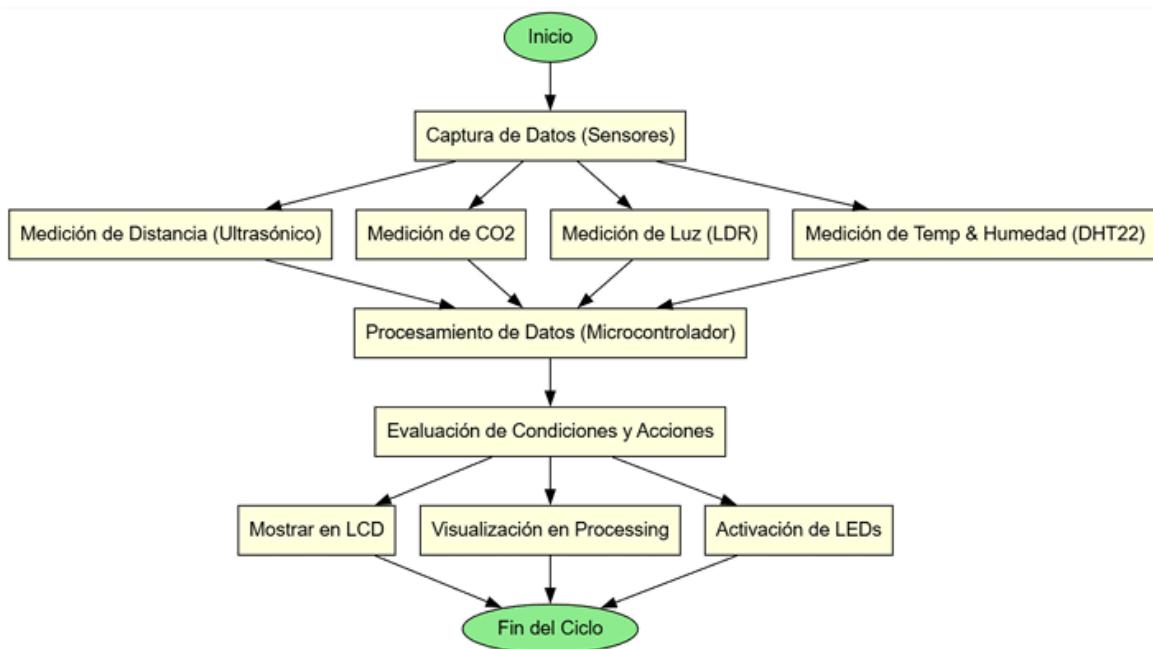
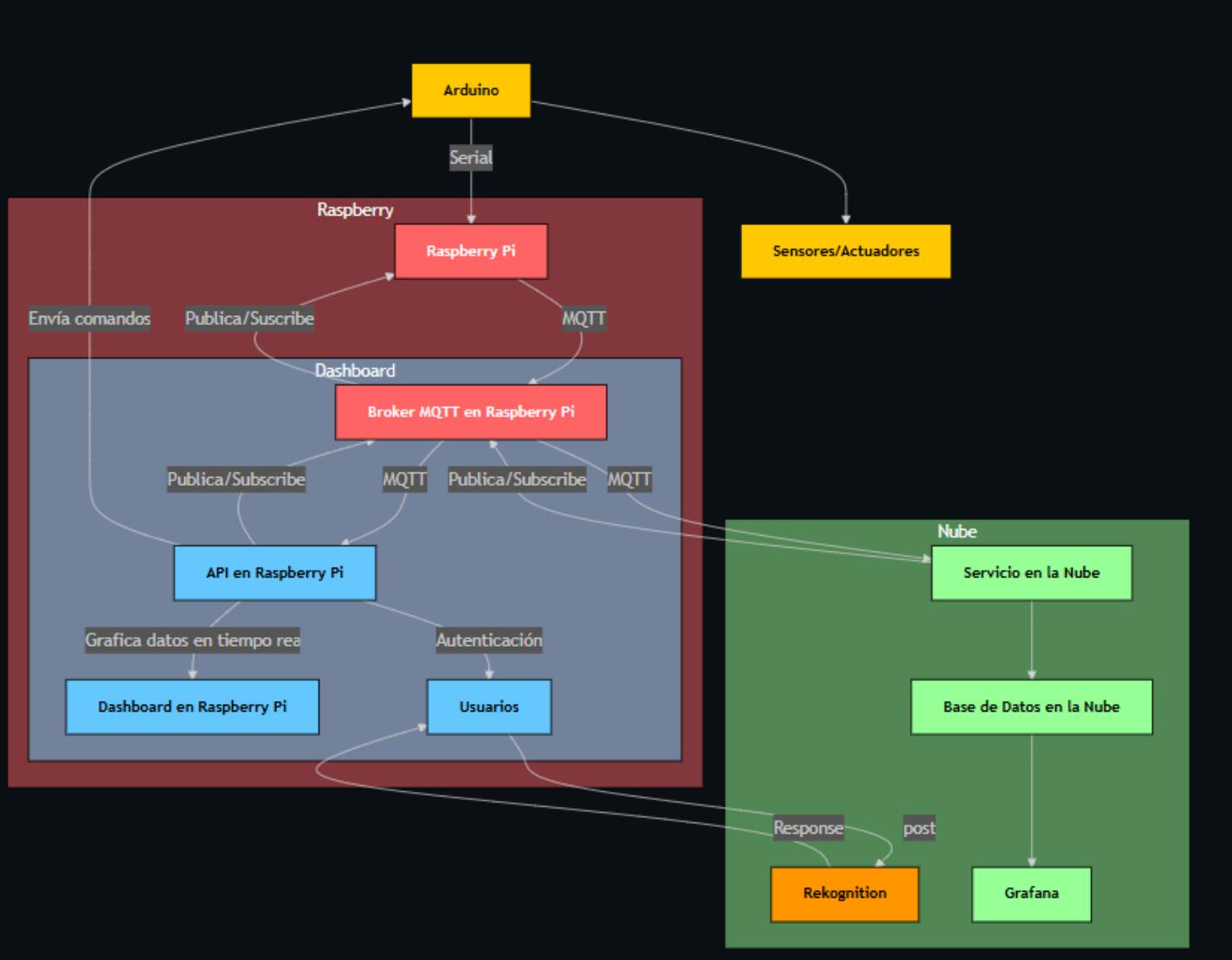


Diagrama Procesos (Fase 1 y 2)



Boceto Fase 3



Link del repositorio

https://github.com/DiegoCali/ARQUI2B_1S2025_G9.git